Introduction to Artificial Intelligence (Belief revision - Group 5)

Authors

Georgios Stamatopoulos - s230251

Mario Rojo Vincente - s230243

Karrar Adam Mahdi - s230432

Bence Kern - s201758

May 1, 2023

Contents

1 Introduction

In this paper, we will explain the process that we followed to design and implement a belief revision agent. For that we considered a belief agent (referred to as BA from now on) to be an entity capable of carrying out logical reasoning. As part of the BA, the belief base (referred to as BB from now on) is a fundamental concept in the area of Artificial Intelligence, particularly when dealing with knowledge and reasoning. The BB is the collection of the beliefs of the agent. The BB includes information about the agent's condition and its environment, which the agent uses as bases for reasoning. The BB supports two operations; expansion and contraction in our project. Using them the agent can update its BB to adapt to changes in its condition and environment. Both these operations will be further discussed in subsequent sections of this paper.

2 Design and implementation of belief base

The design of the belief base was based on the syntax and semantics used by the SymPy library. More specifically :

- conjunction was represented by "&"
- disjunction was represented by "|"
- implication was represented by "»"
- bi-implication was represented by "(A»B) & (B»A)" when A and B are atomic propositions
- negation was represented by "~"
- atomic propositions that compose the beliefs can be represented by almost every letter of the alphabet

Regarding the implementation of the belief base, we concluded to create a class to handle most of the operations in the belief base. The idea of having an object of type Base to represent the belief base makes it possible to create multiple independent belief bases and as a result, to represent logic for different environments. This object has a list of beliefs, in which each belief is represented by its corresponding FunctionClass (from the Sympy library), all of them are stored in their conjunctive form, as well as their priority order (of type Integer). In addition to the BB, the object contains several functions for the actions that can be performed on its BB, as well as functions for testing the AGM postulates and a function for entailment.

To easily interact with the belief base, a simple menu was created in the main file. Then, the logic for the belief base was written in the base.py file, which included the class mentioned

above, helper functions to manipulate and reason about beliefs, as well as test cases to test whether the belief base is operating correctly (e.g.AGM Postulates, entailment tests).

3 Logical entailment

For logical entailment, we have chosen to use the implementation of the resolution algorithm based on the pseudo-code represented as an example in the book Russell et al.. The majority of operations take place in the function called "entailment", where from the belief base we transfer the beliefs into CNF form and split them into sentences by the "&" operator (using a function "SplitByOperator" implemented by us). Then we add the sentence for which we would like to examine whether the entailment happens, using the equation KB & $\neg\alpha$. It has been also made using the above-mentioned "SplitByOperator" function by adding the negation of the new sentence to the already made clauses and making the split using the "&" operator.

Then between each of the clauses created, we call the resolve function, in which the initial clauses are now broken apart using the "|" operator to make sure to iterate through each symbol. The condition checks whether they negate each other. If the condition happens, then these specific symbols are changed to empty strings, and the updated clauses get returned for the main thread of entailment, where they will be the resolvents. The true outcome of entailment depends on whether the resolvents contain the empty clause. Otherwise, all the resolvents are getting saved out to a new list of clauses and the agent checks whether the newly made list is a subset of the original list of clauses.

Since entailment played a major role in the implementation of our agent in revision for both expansion and contraction, it was crucial to creating broad separate test cases for entailment, which are included in the provided code under the "entaiment_tests" function.

In the following figure, we can see a small example of the function in action. When our belief base is "A&B&C" and the sentence in check is "A&B", we should get back "True" from the entailment function since the belief base entails the example sentence. Therefore the received result meets our expectations.

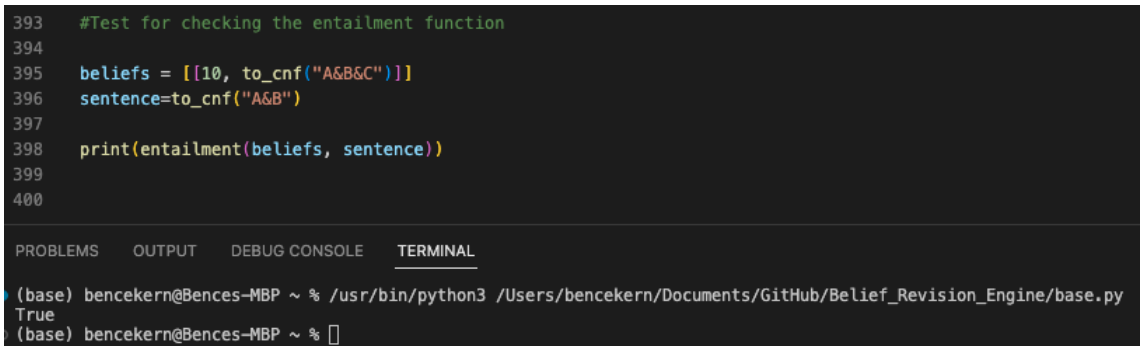

Figure 1: The outcome of the mentioned entailment test

4 Contraction

The basic idea regarding contraction was to use partial contraction and create a maximal subset of the belief base that does not imply the sentence we want to contract. We have used our entailment function to be sure we do not only remove the same sentences but also remove those that implicate them.

All steps for contraction including the checks for the validity of the operation are included in the "revision_contraction" function. At the end of the contraction, we get information on which sentences have been removed based on the propositional logic rules and the orders of the operation and the beliefs, and the belief base gets updated.

To be sure of the correctness of the contraction and its revision, we call the success, the inclusion, and the vacuity AGM postulates. The contraction has been tested for many sentences and belief bases, where not only did we get the expected outputs, but also successfully applied the AGM postulates for testing (all the tests were successful in all test cases). The following figure shows an example test that was made for a use case where the belief base was "A&B", "A&B&C", and "D", all of them with 10 priority orders. The sentence we wanted to remove was "A&B". Not to consider the rankings, it was obvious to remove the "A&B" since this is the sentence that was meant to be removed. However, to avoid also sentences that imply "A&B&C" should be also removed. In the first test case, we set the priority order of the sentence we want to remove 11, to actually test the contraction:

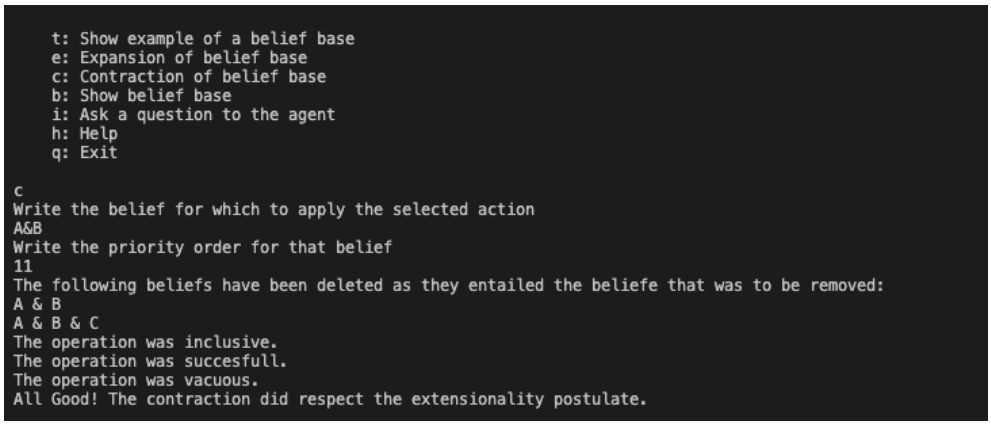

Figure 2: The outcome of the mentioned contraction test with priority order eleven

Then the second test has been made with the priority order of the sentence set to two, to test that in this case, the agent will deny the contraction.

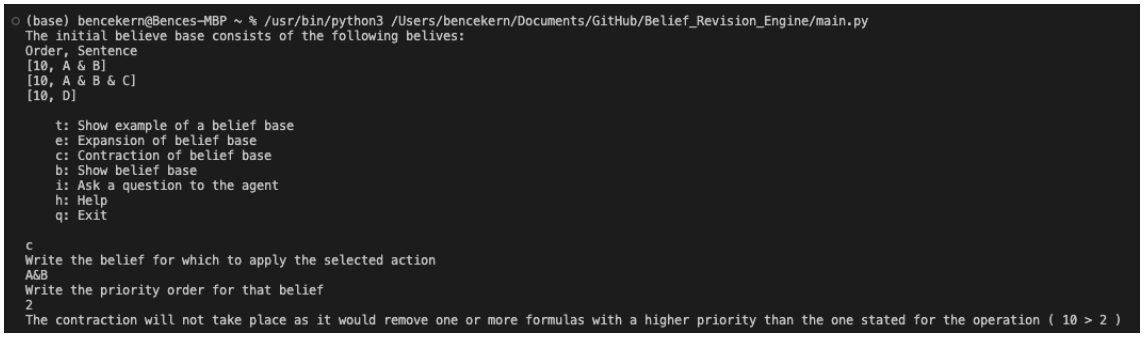

Figure 3: The outcome of the mentioned contraction test with priority order two

Since we do not store the history of belief bases according to previous runs for different sentences, to test whether our agent respects the extensionality postulate, we added a small function called "test_extensionality_contraction". In that, we created two identical BBs and tested whether the final state of both BBs after the operation was equal. Our extensionality postulate for contraction returned a true value, which means our agent passed this AGM postulate. It is also worth mentioning, that all of our AGM postulates for contraction are getting called from the "testContractionAGMPostulates" function, which is executed when the revision regarding contraction was called.

5 Expansion

The idea of the expansion in logical belief revision is adding a new belief to a given belief base, its goal is to add new beliefs to the BB in light of new evidence, while also preserving

the logical consistency of the belief set. This means that any contradiction or inconsistencies between the original belief base and the new belief must be resolved before the operation takes place. Expansion is explained in AGM framework as a sentence $\boldsymbol{p}$ added to original base $\boldsymbol{K}$ with nothing being removed, resulting in a new belief base $\boldsymbol{K} + \boldsymbol{p}$ (K*) which is the smallest logically closed set that contains both $\boldsymbol{K}$ and $\boldsymbol{p}$.

On the "revision_expansion" function we also implemented several checks to ensure the consistency, inclusion, 'vacuity, success, and extensionality of the operation. The expansion operation requires first carrying out some checks on the validity of the operation and then expanding the knowledge base with the new belief given all checks were successful. This is Figure 3 below where we used the AGM postulates such as consistency, inclusion, succession, and vacuity.

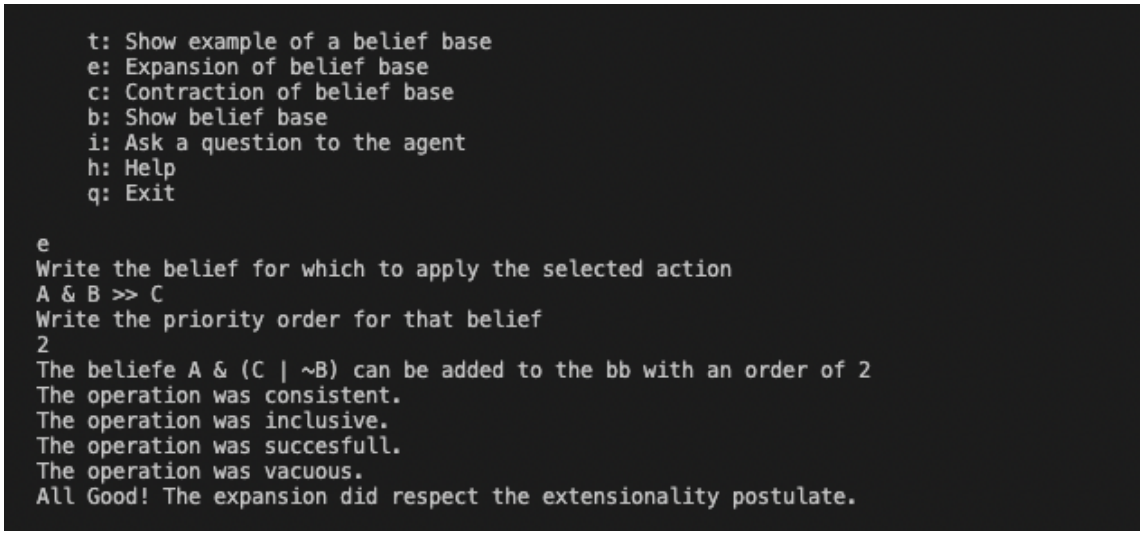

Figure 4: Showing expansion test for operation of 4 AGM postulates are all good

The above test 4 consists of adding the belief "A&B » C" to BB with priority order 2. To make sure all testing criteria were satisfied, we used the AGM postulates as tests for consistency, inclusion, success, vacuity, and extensionality.

From the different postulates; consistency reacquires that the resulting BB does not contain any contradictions, inclusiveness requires that the new belief set is a superset of the original belief set, vacuity requires that the revision operation yields no effect when applied on a sentence that is not entailed by the BB, while success requires that the operation was completed. Finally, we checked if the extensionality postulate is respected by the expansion operation which requires that two equivalent BBs yield equivalent expanded BBs (or BB*) if affected by the same operation.

6 Final thoughts

During this project, we developed a belief revision agent that allowed us to reason with and revise a set of beliefs using logical inference. We learned about 2 different types of logic used in AI, such as first-order logic and propositional logic, and how the latter can be implemented to model reasoning tasks. Our belief revision agent specifically demonstrated how logical inference can be used to revise a set of beliefs in response to new information. This means that we needed a solid understanding of concepts such as entailment, contraction, revision, and expansion, not only because we had to implement the belief revision agent, but also because we had to test whether it functions correctly. Overall, this project highlighted the critical role of logic in AI and its capability to enable intelligent systems to reason and make decisions in complex and uncertain environments.

7 Future work

In the future, we would like to implement the optional element of this assignment (Mastermind) by coding the rules of the game, adding a search algorithm as well as a heuristic, and perhaps making some modifications to the structure of this project. Lastly, we would like to realize a separate implementation of the belief revision engine using first-order logic.

References

S. Russell, P. Norvig (R, and N). *Artificial Intelligence: A Modern Approach (fourth edition)*.